ECE 276A: Project-1 Report

Jaidev Shriram (University of California, San Diego)

***Abstract*— Pose estimation is a critical problem in robotics, such as in SLAM where we must localize a moving body. This is a difficult problem due to the nature of the sensors that are used, which often must be calibrated carefully and even after calibration, optimized to reduce noise that is present in the dataset. Further, we require additional constraints on the problem to In this project, we demonstrate how to calibrate an IMU sensor and use its reading to estimate the orientation of a moving body in 3-D space. We then demonstrate the effectiveness of our poses by constructing a panorama of the scene. We find that our estimated poses are highly accurate across all three dimensions. Further, we find that the panoramas constructed are seamless and continuous, faithfully representing the spatial layout of the scene.**

I. Introduction

A fundamental problem in robotics and computer vision is pose estimation, the task of estimating the position and orientation of a body in 3D space, and updating these estimates reliably as the body moves through space. The clearest example is autonomous navigation, where pose estimates can inform the robot about where it is and subsequently help it plan a trajectory to a predefined destination. However, accurately and reliably estimating the pose of a moving robot is far from easy, due to inaccuracies in sensor readings. For instance, even if we can guess the rough trajectory of a robot using the control inputs given to it, in practice, the actual trajectory will deviate slightly. Over time, these errors can compound, or cause drift, which makes poses unreliable. Hence, in this project, we focus on the task of orientation tracking, a more relaxed version of the pose-tracking problem where a camera is purely rotating in 3D space. To demonstrate the utility of these pose estimates, we then create a panorama of the world captured by the camera.

This project is primarily concerned with estimating the pose of a body in space using IMU data, which records the linear acceleration and angular velocity of the body in three dimensions. Using these values, we can build rough estimates for the pose at each timestamp by simply following the motion model of the system. In addition to this, we perform an optimization step to refine these estimates, described in more detail in section III.

Our results show that the optimization procedure is able to effectively estimate the pose of the 3D body. Finally, we also demonstrate a panorama construction task that leverages these estimated poses.

II. Problem Formulation

There are two fundamental problems for this project: Orientation Tracking and Panorama Stitching, which we will consider separately.

Orientation Tracking. Consider a hand-held camera in an environment that is initially at a pose q_0 at time t_0. Assuming a user rotates that rotates the camera over the next T timesteps, to yield camera poses $\{q_1, \dots, q_T\}$ for timestamps $\{t_1, \dots, t_T\}$. Given just p_0, and sensor readings u representing the angular velocity, ω and acceleration, a, for the timesteps $[1, T]$, the task is to estimate $q_{1:T}$.

Panorama. Consider a set of poses, $\{q_1, \dots, q_T\}$, and the corresponding images captured from these, $\{I_1, \dots, I_T\}$. We must then combine these images into one image, I, such that I represents a wide-angled continuous view of the scene, with multiple images blending seamlessly into one, while respecting the spatial layout of the scene.

III. Approach

A. Orientation Tracking

1) IMU Calibration: The first step to estimate the orientation of the body is to convert the raw IMU data to relevant units. We do this by following the instructions in the IMU instruction manual. The IMU packet provides a $6 \times N$ matrix, holding six values for N timestamps, with the six values corresponding to $A_x, A_y, A_z, W_z, W_x, W_y$ respectively. These are however the raw A/D values which we convert to physical units as follows:

$$V_{phys} = (V_{raw} - V_{bias}) * \text{scale}, \tag{1}$$

where V_{phys} refers to the value in physical units, V_{raw} refers to the raw sensor values, V_{bias} refers to the bias that appears when the physical value should correspond to zero, and the scale is calculated as:

$$\text{scale} = \frac{V_{ref}}{\frac{1023}{\text{Sensitivity}}} \tag{2}$$

For this project and given IMU, the V_{ref} is 3300 mV, and sensitivity is specified in the manual for the gyroscope and accelerometer separately.

2) Estimating the Orientation: We will consider the orientation estimation process in two parts:

Motion Model: Let f represent a function that describes the motion model of the robot, predicting q_{t+1} given q_t and u_t. We can define f as follows:

$$q_{t+1} = f(q_t, u_t) = q_t \circ \exp([0, \tau_t \omega_t / 2]), \tag{3}$$

where ω_t represents the angular velocity given in the IMU data u_t and τ_t represents the time elapsed between timestep t and $t+1$.

Observation Model: Let h be a function that transforms gravitational acceleration to the current body frame, which we cover in more detail later. We know that the body is undergoing pure rotation (prior information from the dataset), due to which we can say that the acceleration a_t is expressed as:

$$a_t = h(q_t) = q_t^{-1} \circ \begin{bmatrix} 0 & 0 & 0 & -g \end{bmatrix} \circ q_t. \tag{4}$$

Hence, the optimal set of poses $q_{1:T}$ is then those q_i that satisfy both $f(q_i, u_i)$ and $h(q_i)$ the best. Alternatively, the poses $q_{1:T}$ must minimize the cost function c as:

$$c(q_{1:T}) := \frac{1}{2}\sum_{t=0}^{T-1} \|2\log(q_{t+1}^{-1} \circ f(q_t, \tau_t, \omega_t)\|_2^2 + \frac{1}{2}\sum_{t=1}^{T} \|a_t - h(q_t)\|_2^2 \tag{5}$$

However, quaternions must satisfy the property that they have a unit norm, due to which we can form our constrained optimization as:

$$\begin{aligned} &\min_{q_{1:T}} c(q_{1:T}) \\ \text{s.t.} &\|q_t\|_2 = 1, \qquad \forall t \in \{1, 2, \ldots, T\} \end{aligned} \tag{6}$$

Note that the first term in the cost function calculates the relative rotation between the predicted orientation $f(q_t, u_t)$ and q_{t+1}, which when converted to axis-angle representation via the log map, should have a zero norm. The second term simply minimizes the difference between the ground truth acceleration a_t and derived acceleration $h(q_t)$.

Implementation Details. We set $q_0 = [1, 0, 0, 0]$ and run gradient descent for 5000 epochs with learning rate 0.01 using 'jax'. To ensure that each update step retains a unit form for each quaternion, we use a modified update step:

$$q_{t+1} = \frac{q_t - \alpha \nabla c(q_{1:T})}{\|q_t - \alpha \nabla c(q_{1:T})\|} \tag{7}$$

B. Panorama

We generate the panorama by inscribing all images I_i on a sphere of unit radius, using the pose q_i to determine the position of the image on the sphere. This sphere is then mapped onto a rectangular image.

Inscribing Image on the Sphere. We use the spherical coordinate system to inscribe the images on a sphere (Fig. 1), where the longitude λ and latitude ϕ of a pixel (x, y) are calculated as follows:

$$\begin{aligned} \lambda &= -\frac{\text{Horizontal FOV}}{W} \cdot \left(y - \frac{HorizontalFOV}{2}\right) \\ \phi &= \frac{\text{Vertical FOV}}{H} \cdot \left(x - \frac{VerticalFOV}{2}\right) + \frac{\pi}{2} \end{aligned}$$

Hence, the cartesian coordinates for an image can be calculated as:

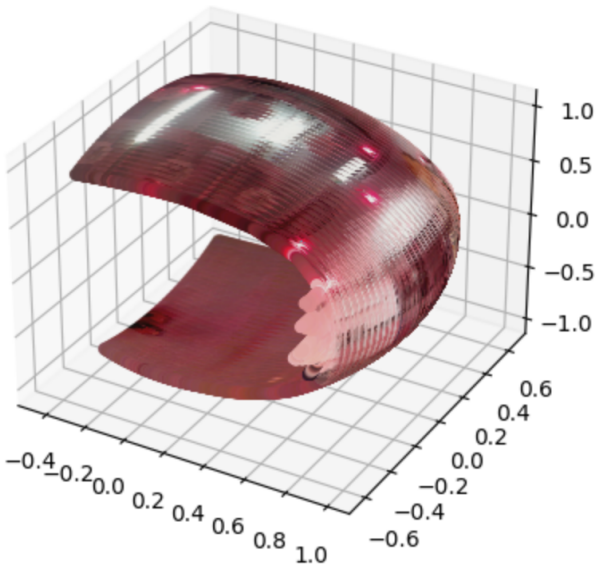

Fig. 1: The first step in creating a panorama is inscribing the images on a sphere of unit radius, as visualized here.

$$\begin{aligned} x &= \cos(\lambda)\sin(\phi) \\ y &= \sin(\lambda)\sin(\phi) \\ z &= \cos(\phi) \end{aligned} \tag{8}$$

We can now rotate these points from image I_i to the world frame $\{W\}$ using R_i, which is the rotation matrix corresponding to quaternion q_i.

Map Projection. Once we have our points on a sphere, we can now recover the spherical coordinates for all points on this sphere as follows:

$$\begin{aligned} \phi &= \cos^{-1}(z) - \frac{\pi}{2} \\ \tan(\lambda) &= \frac{y}{x}, \quad \phi \in [0, 2\pi] \end{aligned}$$

With these spherical coordinates, we can simply map them back to a rectangular image using the inverse of the inscription operation done earlier, where the pixel coordinate (x', y') is calculated as:

$$\begin{aligned} y' &= \frac{\lambda}{2\pi} * \text{ WIDTH} + \frac{\text{WIDTH}}{2} \\ x' &= \frac{\phi}{\pi} * \text{ HEIGHT}, \end{aligned} \tag{9}$$

where x', y' are rounded to the nearest integer.

IV. Results

In this section, we present some results for the orientation tracking and panorama, for the trainset and test set. The only difference with the testset is the lack of a ground truth to compare the results.

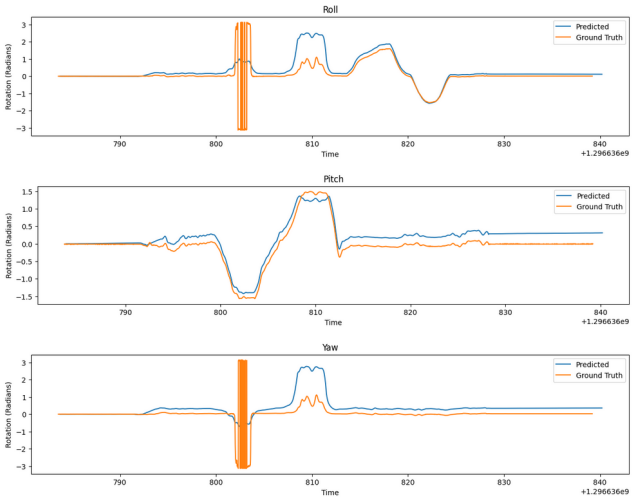

Fig. 2: Predicted poses using just motion model vs. ground truth poses

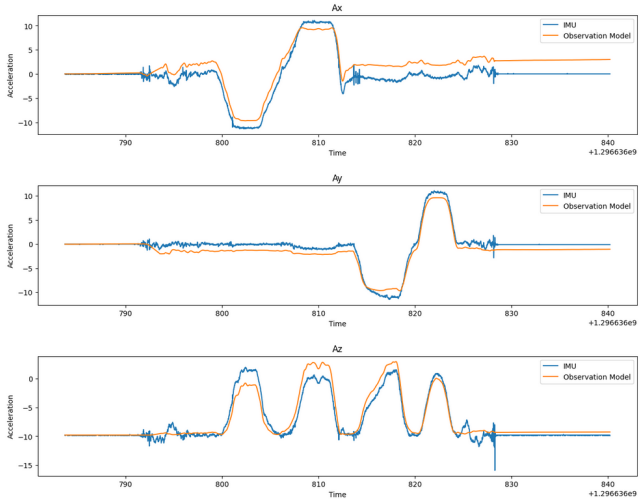

Fig. 3: The acceleration computed by the observation model is close to the IMU estimates even before optimizing.

A. *Orientation Tracking*

Images 5-15 show the results for orientation tracking.

B. *Panorama*

The results of panorama creation can be seen in figures 16-20.

V. Discussion

A. *Orientation Tracking*

IMU gives rough pose estimates. As seen in figure 2, using just the motion model to create our poses from the IMU data yields poses that are close to the ground truth. However, when the camera begins turning, we notice some differences. As seen in the results section, optimizing these poses with the added observation model helps refine the poses.

The unoptimized poses largely respect the observation model. As seen in figure 3, the acceleration values calculated using the observation model is close to the values that come from the IMU data.

The first 100 timestamps tend to be constant. In order to calibrate the IMU and obtain the bias, we compute the mean of the first 100 readings, which tend to be constant across all datasets.

The optimization converges quickly. We notice that the optimization process converges quickly for all datasets in under 500 steps. We train for 5000 epochs, just to obtain the best results. Due to batched optimization in Jax, we are able to finish training in a minute.

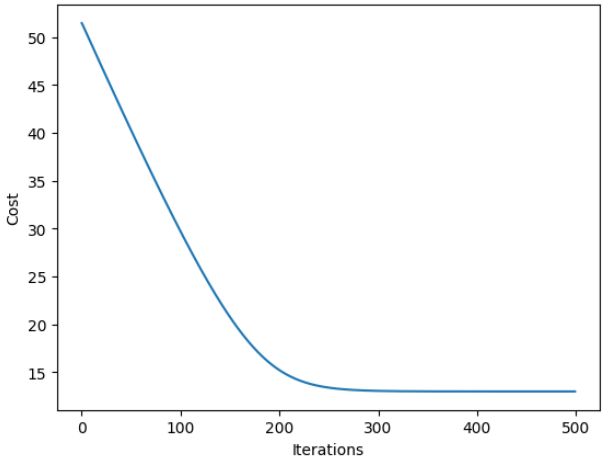

Fig. 4: The optimization converges in under 500 iterations for most datasets

B. *Panorama*

Visualizing the 3D point cloud helps resolve pose errors. We found that visualizing the 3D sphere with inscribed images helped validate the poses and pipeline greatly. This is evident in figure 1.

Downsampling introduces black lines in the image. During early experiments, we created the point cloud using down-sampled point clouds, which caused black lines to appear on the image. While using the entire point cloud did use more memory, it was clearly the optimal solution.

Quick Motion can cause a strange panorama. In the final test scene, we observe extremely rapid movement in the camera, which makes the camera seem like it jumped between poses. This is evident in the resulting panorama as well, as it is not as continuous as the previous panoramas, which give a more complete picture of the

VI. Conclusion

In this project, we describe a technique to estimate the poses of a rotating body in 3-D using just IMU data. Our experiments show us the importance of including the observation model in addition to the motion model, which together can describe a cost function that when minimized optimizes the poses. Finally, we demonstrate the effectiveness of the pose estimation pipeline by creating a panorama. The resulting panorama is continuous and captures the spatial layout of the scene effectively.

A limitation of the current work is the lack of image features used in the pose estimation pipelines. It is likely possible to improve the pose estimates of the body by matching features across images. Further, the panorama created does not currently use colour averaging, and has some discontinuities, which can be fixed in future work.

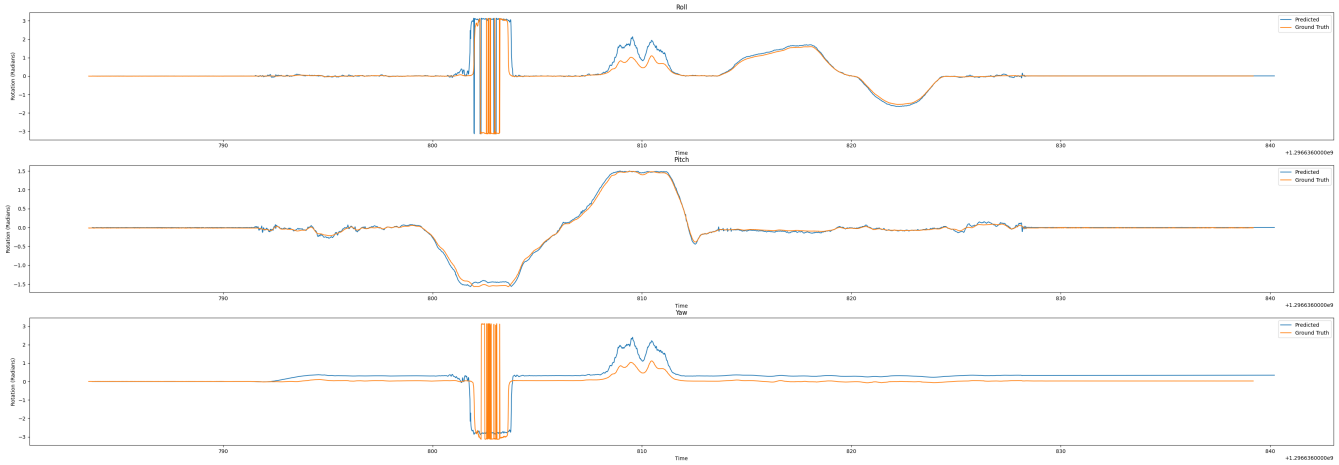

Fig. 5: Predicted Orientation for Scene #1

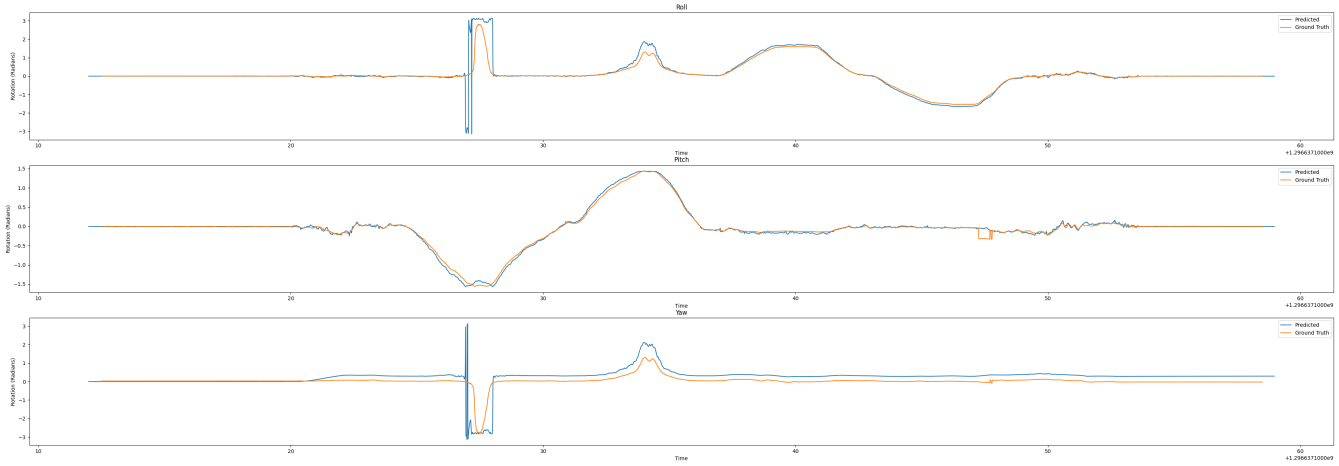

Fig. 6: Predicted Orientation for Scene #2

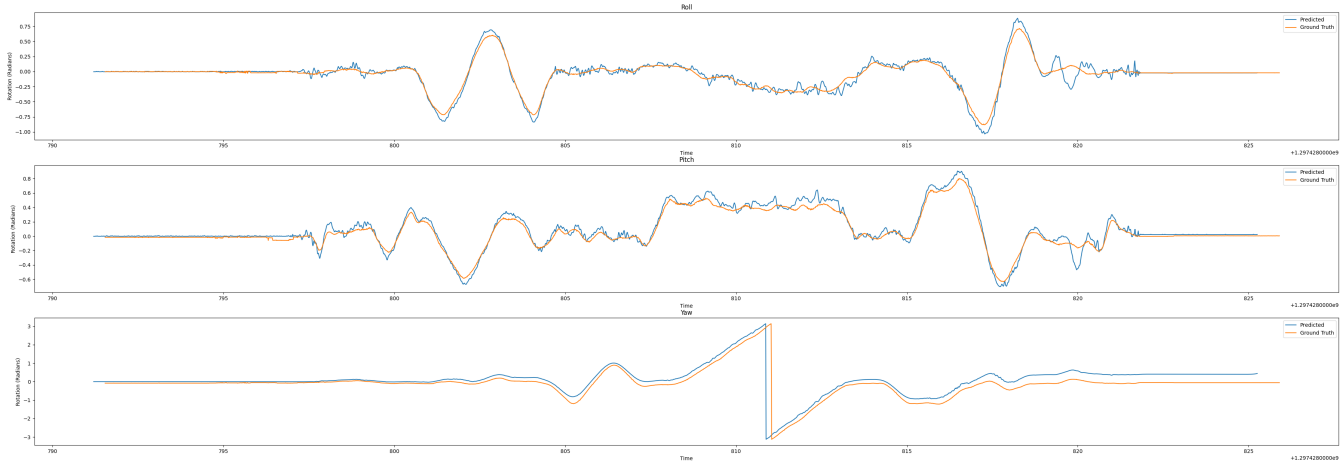

Fig. 7: Predicted Orientation for Scene #3

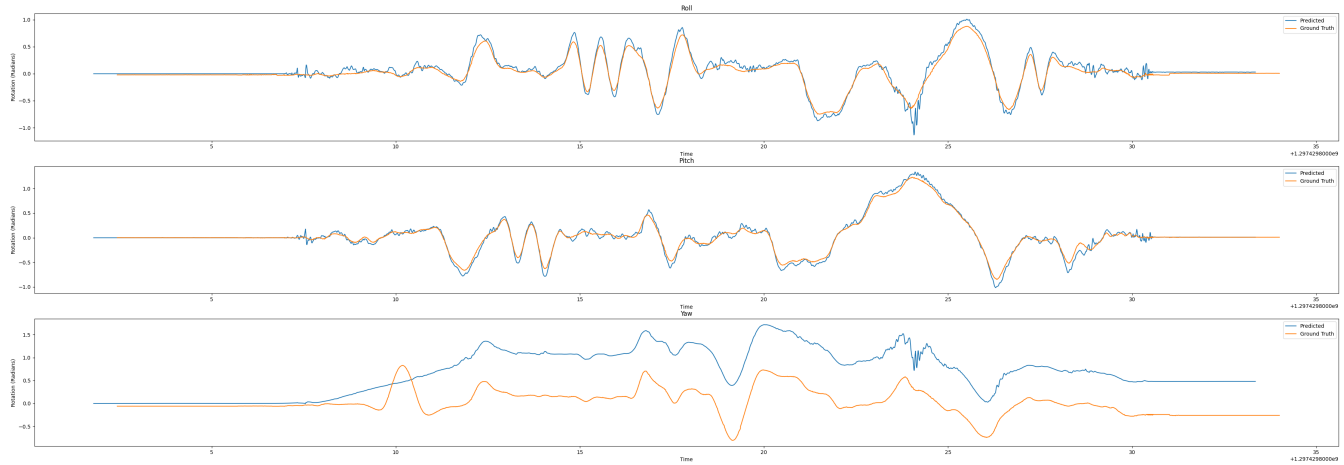

Fig. 8: Predicted Orientation for Scene #4

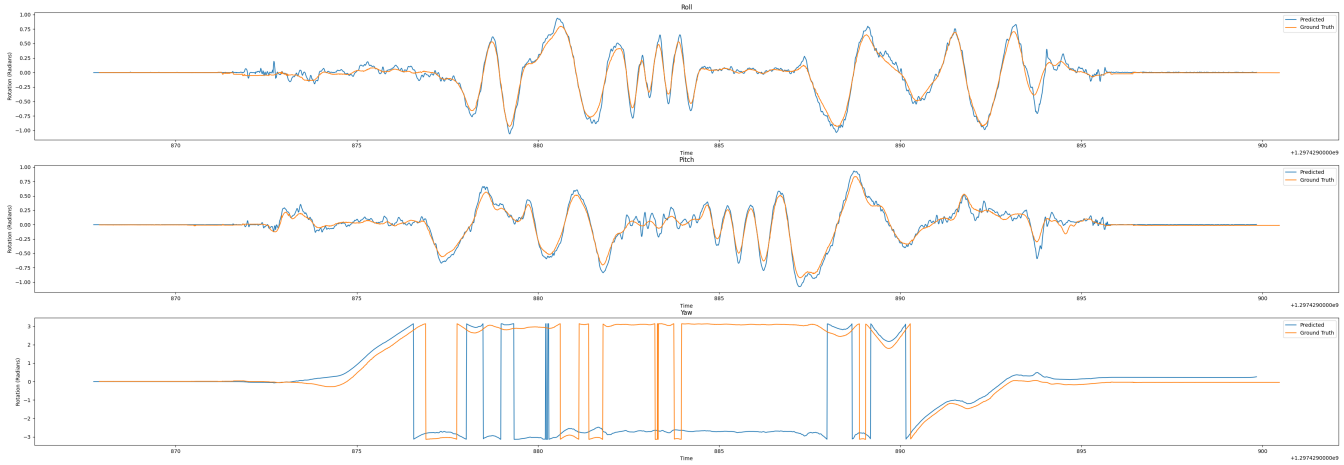

Fig. 9: Predicted Orientation for Scene #5

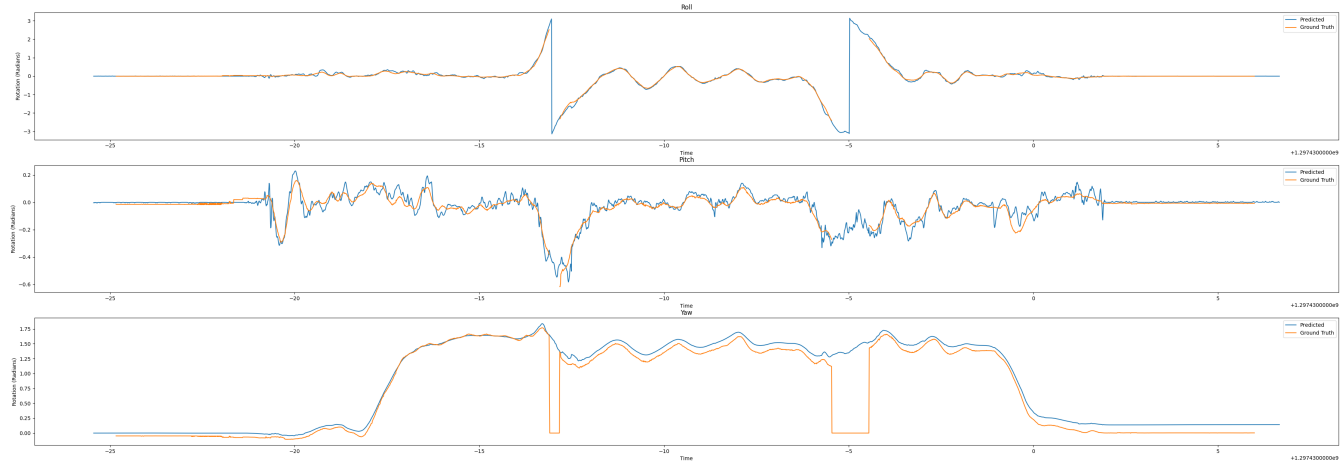

Fig. 10: Predicted Orientation for Scene #6

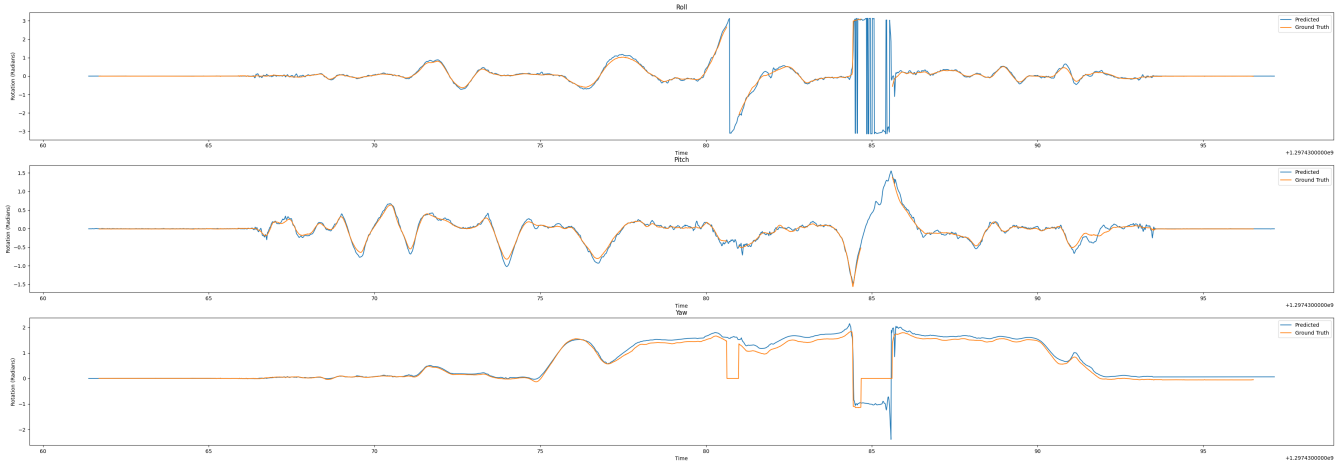

Fig. 11: Predicted Orientation for Scene #7

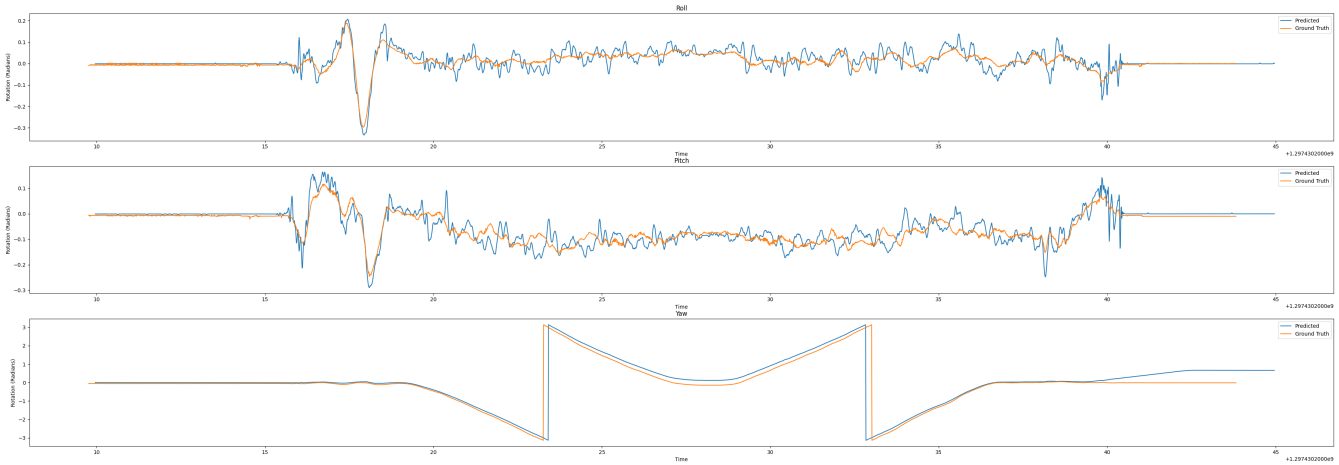

Fig. 12: Predicted Orientation for Scene #8

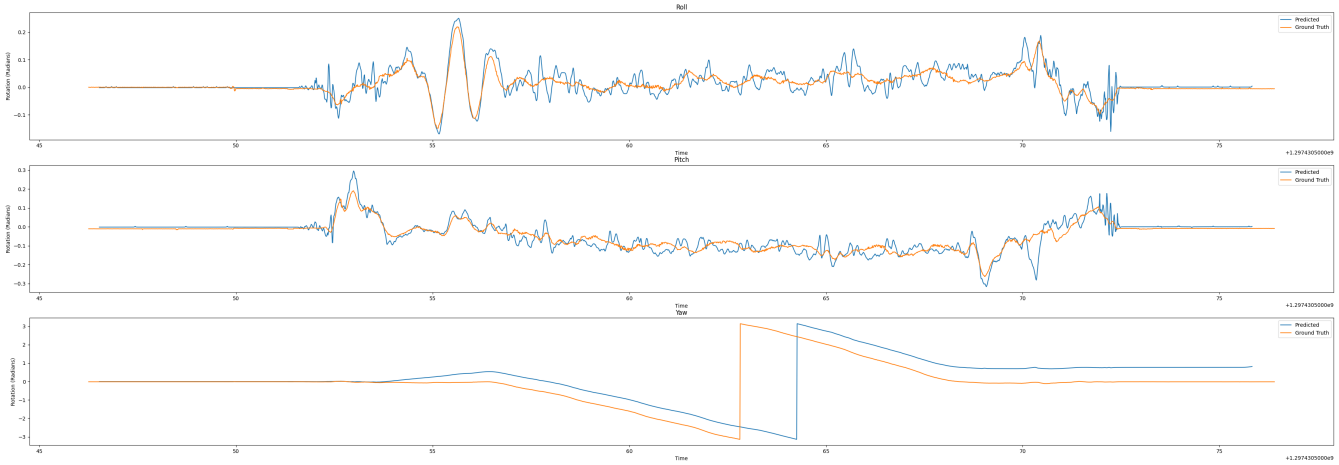

Fig. 13: Predicted Orientation for Scene #9

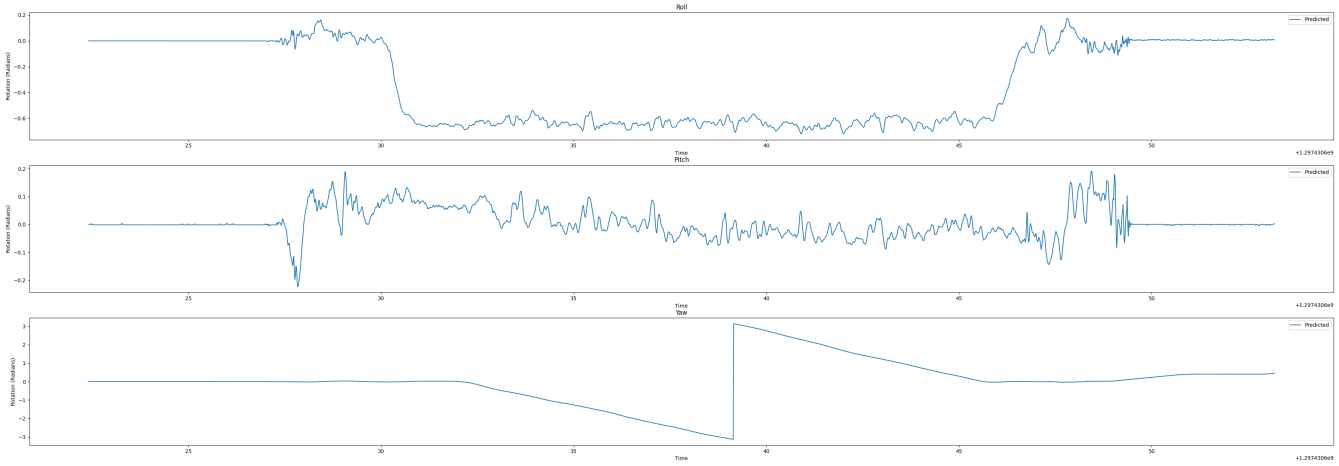

Fig. 14: Test: Predicted Orientation for Scene #10

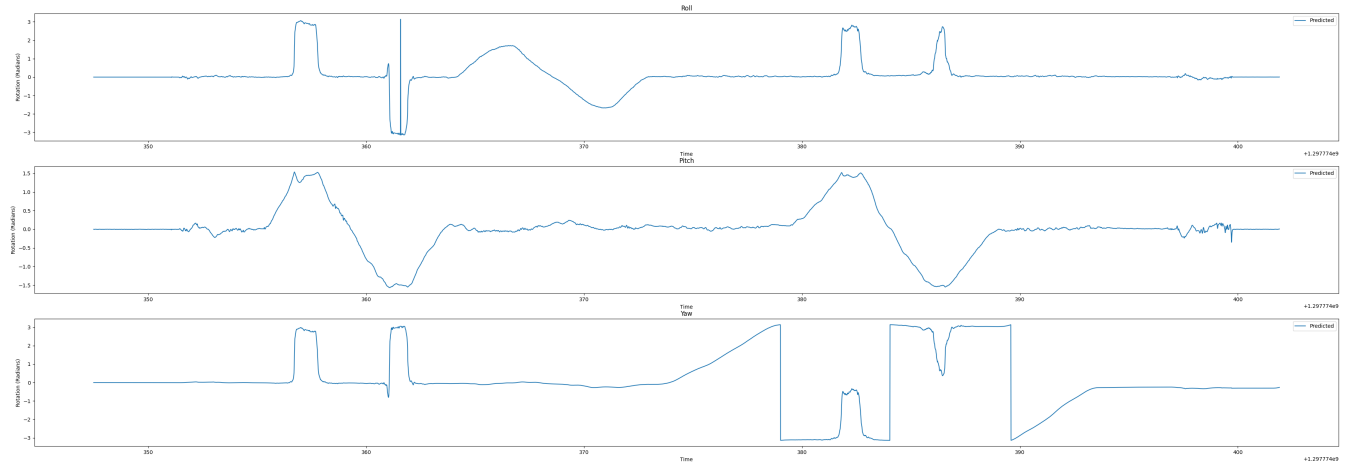

Fig. 15: Test: Predicted Orientation for Scene #11

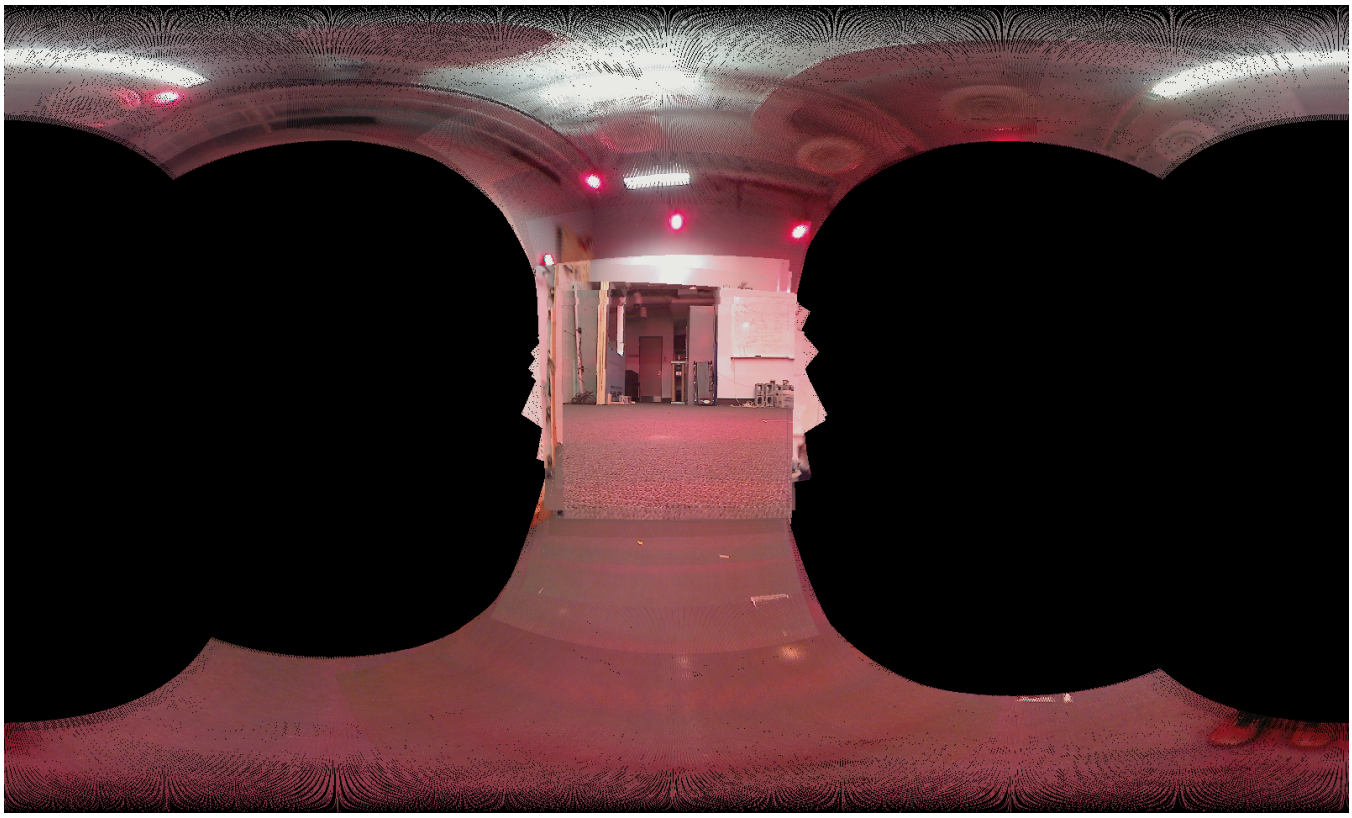

Fig. 16: The stitched panorama for scene #1

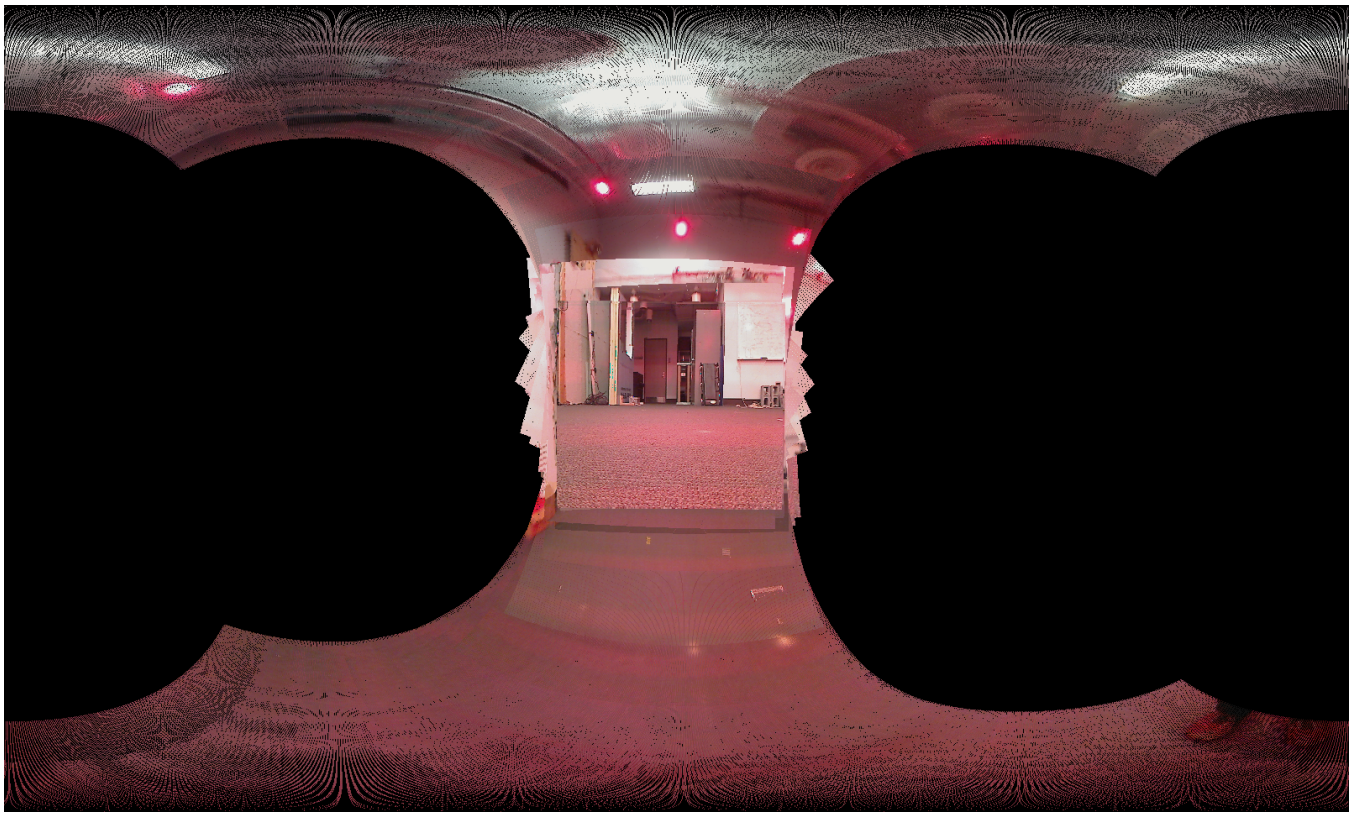

Fig. 17: The stitched panorama for scene #2

Fig. 18: The stitched panorama for scene #8

Fig. 19: The stitched panorama for scene #10

Fig. 20: The stitched panorama for scene #11